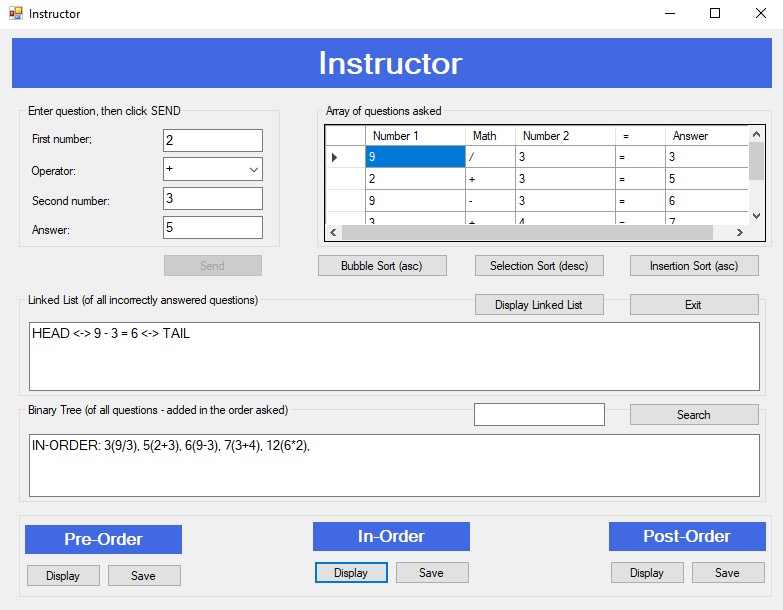
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Student Name | | Asami Paddison | Student Number | 456657238 |
| Unit Code/s & Name/s | | ICTPRG523 Apply advanced programming skills in another language | | |
| Assessment Type | | Portfolio of Evidence | | |
| Assessment Name | | Networked Math Quiz Apps | Assessment Task No. | AT1 |
| Assessment Due Date | | Thu 02-Jun-2022 | Date submitted |  |
| Assessor Name | |  | | |
| **Student Declaration:** I declare that this assessment is my own work. Any ideas and comments made by other people have been acknowledged as references. I understand that if this statement is found to be false, it will be regarded as misconduct and will be subject to disciplinary action as outlined in the TAFE Queensland Student Rules. I understand that by emailing or submitting this assessment electronically, I agree to this Declaration in lieu of a written signature. | | | | |
| Student Signature | Asami.P | | Date | 2/6/2022 |
| **PRIVACY DISCLAIMER:** TAFE Queensland is collecting your personal information for assessment purposes. The information will only be accessed by authorised employees of TAFE Queensland. Some of this information may be given to the Australian Skills Quality Authority (ASQA) or its successor and/or TAFE Queensland for audit and/or reporting purposes. Your information will not be given to any other person or agency unless you have given us written permission or we are required by law. | | | | |

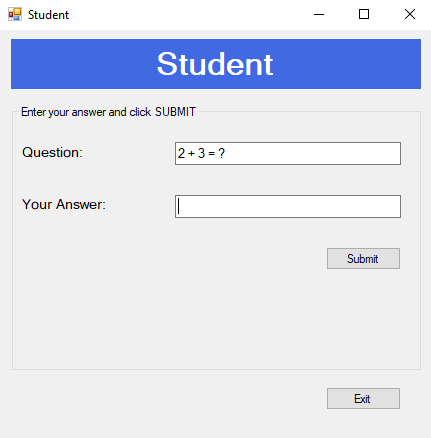
|  |  |
| --- | --- |
| Instructions to Student | **Learning Support**  Additional support is available to help you achieve your learning goals. Speak to your teacher or a Learning Skills Centre team member if you feel that you may benefit from some extra support. The Institute provides extra support through the Disability Support Unit and the Learning Skills Centre.  RPL (Recognition of Prior Learning) is available for this unit. Speak to your teacher/assessor to check if you qualify for RPL.  **Conditions of Assessment**  You will need to complete the learning and undertake all assessments satisfactorily to be deemed competent. You are responsible for complying with all assessment item instructions; submission and collection requirements; undertaking assessment tasks honestly and retaining a copy of all assessment items.  You must submit assessment items by the due date, unless an extension has been granted by your teacher. Failure to submit assessment items by the due date will result in a “did not submit” being recorded and depending on your circumstances, you may be granted one final resubmission.  To be judged competent in this assessment item the student is required to demonstrate competence in all indicators shown in the marking guide.  **The Classroom as a Simulated Work Environment**  Students must be aware and take responsibility for the problems of working in a shared IT environment. Problems such as noise levels, production flow, interruptions and time variances are common to workplaces. In the simulated environment provided in the classroom these problems can take the form of:   * Other students who continually ask questions or talk aloud while thinking * Fire drills, projector not working, printers running out of paper or toner cartridge * Miscalculating how much work you can do in one day, missing classes and so on.   Some things are unavoidable and you must devise strategies to overcome them, for example, we cannot stop students from asking questions or entering at exiting the class. Other things are unpredictable (e.g. fire drills). You need to be aware and plan and organise your work allowing some extra time for unavoidable and unpredicted events. |
|  | **Assessment Criteria:**  To achieve a satisfactory result, your assessor will be looking for your ability to demonstrate key skills/tasks/knowledge to an acceptable industry standard.  Refer to the marking criteria document for a detailed list of items.  **Number of Attempts:**  You will receive up to two (2) attempts at this assessment task. Should your 1st attempt be unsatisfactory (U), your teacher will provide feedback and discuss the relevant sections / questions with you and will arrange a due date for the submission of your 2nd attempt. If your 2nd submission is unsatisfactory (U), or you fail to submit a 2nd attempt, you will receive an overall unsatisfactory result for this assessment task. Only one re-assessment attempt may be granted for each assessment task, with the exception of Apprentices or Trainees who are permitted an additional supplementary assessment. **For more information, refer to the Student Rules.** |
| Submission details | Submit your assessment (zip archive of complete Visual Studio C# project with completed file: ICTPRG523\_AT1\_Supporting\_Documentation.docx and this document with your details: ICTPRG523\_AT1\_PE\_TQM\_v2.docx) to the allocated dropbox in **Connect** or to the allocated network folder.  Your teacher will provide all the details for the submission system or network.  Your assignment must be saved with your surname\_student number\_unit/cluster\_AssessmentNumber. For example:   * surname\_1234567890\_ICTPRG523\_1   For re-submissions, an “R” must be added to the file name. For example:   * surname\_1234567890\_ ICTPRG523\_1\_R   The Marking Criteria Sheet must be signed and submitted with your work. |
| Instructions for the Assessor | To be judged competent in this assessment item the student is required to demonstrate competence in all indicators shown in the marking guide and need access to:   * The software development environment * Technical documentation |
| Note to Student | An overview of all Assessment Tasks relevant to this unit is located in the Unit Study Guide. |

# Instructions to Students

You have been asked by a group of teachers to build a network-based Math Challenge Game. The functionality is provided in the prototype they have prepared.



*Instructor application containing required GUI controls*



*Student (client) application containing required GUI controls*

**Instructions are as follows:**

|  |  |
| --- | --- |
| The **Instructor** enters a basic math question. Possible operations include: + (addition)  - (subtraction)  x (multiplication)  / (division)  Using integers (whole numbers)  For instance: 5 + 8 =, 9 / 3 = … |  |
| The **Instructor** then clicks on the **Send**  button. The answer is automatically calculated and displayed in the Answer text field. The entire question is sent to the student application. The **Send** button then becomes inactive.  The **student** who is on another networked computer sees the question appear in their **Question:** text field (but not with the answer) |  |
| The **student** enters the numeric answer into the **Your Answer:**text field.  The **student** clicks the **Submit**  button so the student program can compare the **student’s** answer with the correct answer.  A popup message box appears and indicates to the student whether their answer is correct or not correct.  The **student’s Submit** button would become disabled and a text message is sent to the instructor application – “y” (if answered correctly) or “n” (if answered incorrectly). |  |

|  |  |
| --- | --- |
| All math questions sent to the student are stored within different data structures utilised by the Instructor program:   * **All** math questions are stored in a **List** for the purpose of sorting the elements. The content of the List is displayed on the top right of the **Instructor’s** screen in a **table**.   Three different sort algorithms (choose between Bubble Sort, Selection Sort, Insertion Sort, Merge Sort and Quick Sort) are to be implemented and can be run via the **Sort** buttons - so that the converted array can be sorted. *(For visual effect, you may wish to have one sort in ascending order, second sort in descending order, and third sort in ascending order.)*  You must implement your own methods to do the sorting (and not use the Sort() method within the List class. You can convert the List to an array for this purpose if desired.   * All **incorrectly answered questions** are to be added to a **doubly linked list** (LinkedList in C# and Java) . The content of this doubly linked list is presented in a text area in the centre of the **Instructor’s** screen. Note: The Instructor program receives a “y” or “n” string from the Student program to indicate whether the question was answered correctly or incorrectly. * **All questions** are added into a **binary tree,** sorted by their respective answers. The content of this binary tree can be displayed in a text area at the bottom of the **Instructor’s** screen. Buttons on the bottom of the screen allow the content of to be displayed in Pre, Post or In–Order and to be saved in corresponding text files.   Include code for **searching** a question in either the binary tree or a C# **HashSet** data structure.  The program requires a hashing functionatlity – this can be a **hashing algorithm** set up to output each of the pre, post and in-order traversal displays, or a hash-based search for a question string in the HashSet data structure. |  |

|  |  |
| --- | --- |
| **ERROR-CHECKING**  The following checks must be implemented in the project:  **INSTRUCTOR:**  If the SEND button is clicked and either or both number fields had no entries, then a suitable error message is presented  If the SEND button is clicked and either or both number fields contained non-numeric entries, then a suitable error message is presented.  If the DISPLAY LINKED LIST button is clicked and no questions were sent, then a suitable message should be displayed.  If any DISPLAY button is clicked for PRE-ORDER, IN-ORDER or POST-ORDER options for the Binary Tree traversals, and no questions were sent, then a suitable message should be displayed.  **STUDENT:**  If the SUBMIT button is clicked, and no answer is entered, then a suitable message should be displayed.  If the SUBMIT button is clicked, and a non-numeric answer is entered, then a suitable message should be displayed. |  |

Other error-checking considerations:

* Clicking any of the sort buttons if no questions were sent
* Clicking the Display Linked List button after questions were sent (but all correct – no incorrect ones)
* Clicking the Search button when the search field is empty or if the search entry is not in the correct format

**Programming Obligations:**

In relation to this project, you are required to implement and demonstrate clear competence in the following aspects of programming:

1. Design and build the application based on the specifications on the previous pages with appropriate low coupling and high cohesion, and object-oriented programming techniques wherever possible. Outline / present a procedure for the development of your application. (Note that this procedure needs to account for this being developed in and for a GUI environment.). Include testing and debugging issues.
2. Appropriate design and application of the following data structures:

* List of objects to allow sorting
* Doubly Linked List (LinkedList) of objects to store incorrectly answered questions
* Binary Tree of objects to facilitate traversal methods and/or searching
* Hash set or table of objects to facilitate searching

Include testing and debugging issues.(Elements 1, 2, 4, 5, 6, 7 & 8)

3. Appropriate implementation and application of **three (3) common sort algorithms**.  
Choose 3 of the following: Bubble, Selection, Insertion, Merge and Quick Sort algorithms.

(Elements 2, 5, 6, 7 & 8)

4. Appropriate implementation and application of **program communication**:

* **inter-process communication** through at least one mechanism.
* **operating system 'signals'** to be captured and responded to.(Elements 3, 4, 6, 7 & 8)

5. Utilise a third-party library in the construction of your application indicating how and where you have used the library, and how and where you have referenced the third-party library’s documentation. For instance, you might locate and utilise a third party library that:

* provides a suitable binary tree class (provided for this assessment)
* assists with the graphical presentation or analysis of your doubly linked list, binary tree and/or hashing algorithm.

Include testing and debugging issues.

6. Clear evidence of appropriately planned and structured **testing** and **debugging**. (Element 7)

7. Appropriate internal and external technical **documentation:** (Element 6)

* add **code headers** presenting section headings, author, version control, etc.
* add **useful comments** throughout your code that are likely to prompt others as to the purpose and focus of each code segment and/or to remind you of your current thinking / logic at a later time.
* generate applicable **external documentation** such as UML class diagram and programming documentation
* *outline in detail and provide evidence of how you have managed version control throughout the development of your project solution.*

8. Maintain appropriate ongoing communication with your manager and client *(including emails)* to ensure you stay on track and receive and action client and manager feedback. The technical jargon and detail within your emails needs to be appropriate to your recipients. Specifically, you would be required to:

1. Email your manager before the commencement of this project to secure the project specification.
2. Prepare an email to your manager and then (separately) to your client regarding your proposed third party library, notifying them about the associated licence agreement. The client will need to understand the benefit(s) of the inclusion of this library in their project.
3. Prepare a closing email to your manager and then (separately) to your client regarding the success of your project development endeavours. You should suggest any areas of concern or risk to your manager for inclusion in the development team’s risk matrix. (Element 6)